Quad Lab Proposal for Fundamental Cross Architecture Multi-Level Memory Support

DOE Center of Excellence Performance Portability Meeting

Los Alamos NATIONAL LABORATORY EST. 1943

AWE

Sandia National Laboratories

Lawrence Livermore National Laboratory

April 20th, 2016

Ian Karlin for Quad-lab Working Group

LLNL-PRES-690243

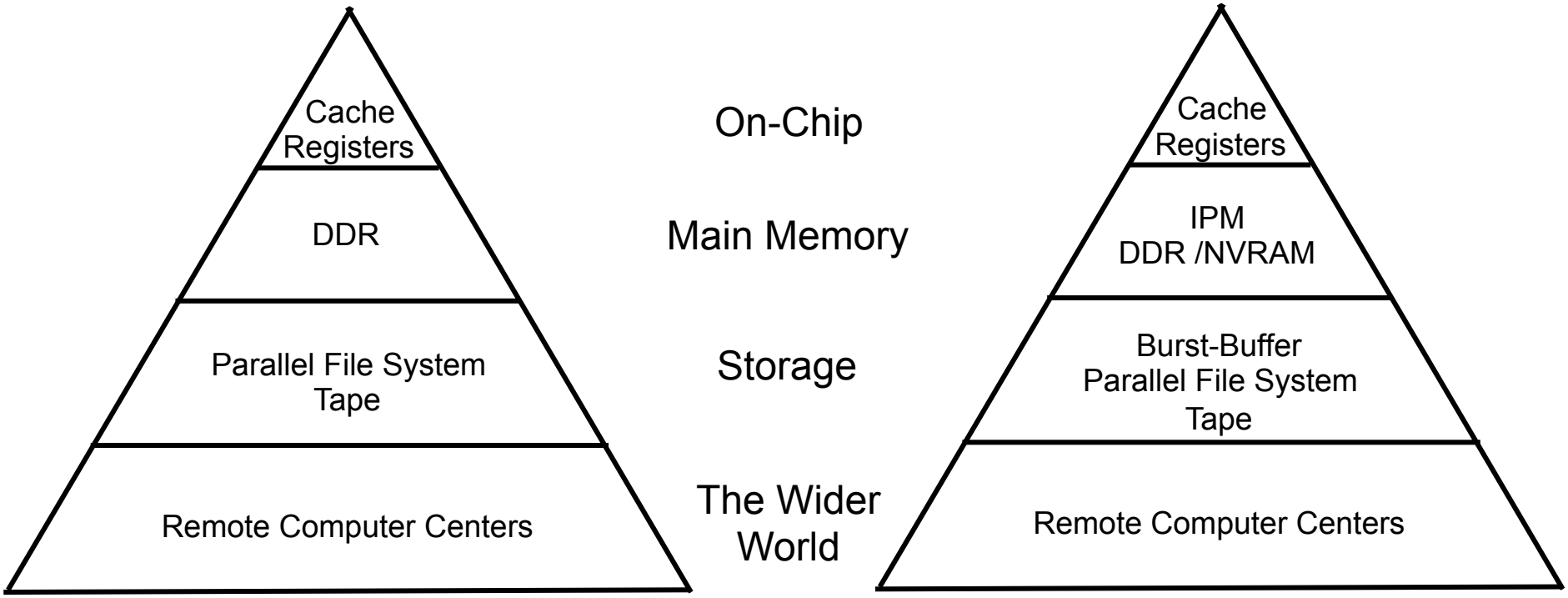
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

How Most of Our Programmers View Today's Memory Systems

Easy to Program

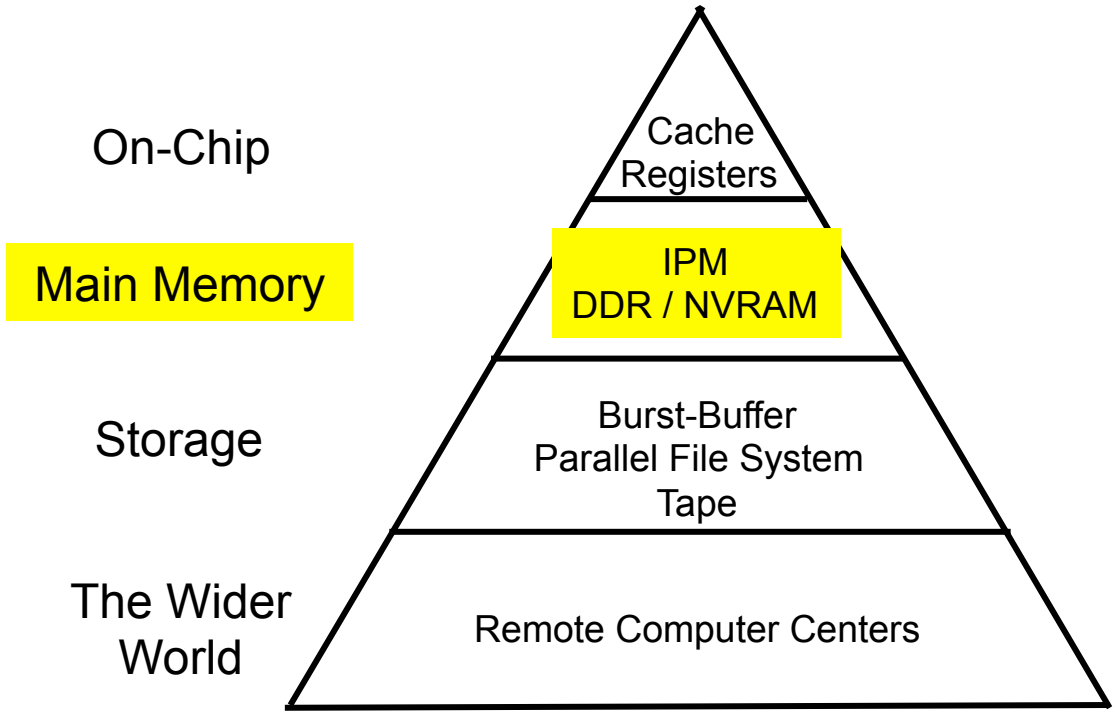
However, Memory Hierarchy's are Deep and Getting Deeper



Today's Heirarchy

Coming Soon to a Center Near You

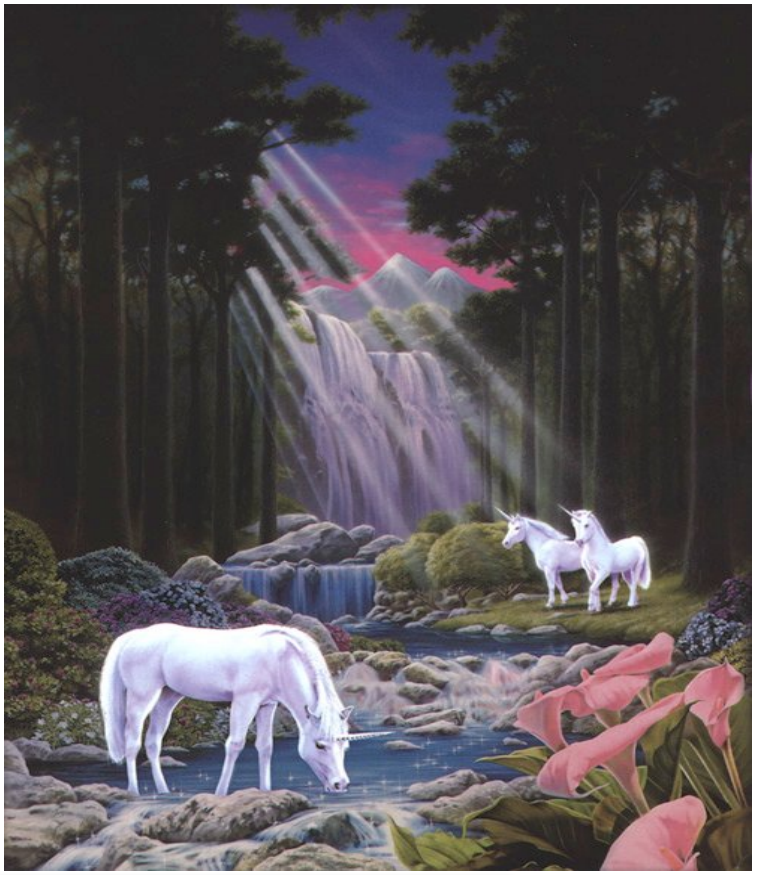
Our Focus is On the Main Memory Changes



Disruptive Throughout the Program and Essential To Use Efficiently For Performance

Vendors Have Hardware/OS Solutions

- Data motion handled for the programmer
  - Cache Modes
  - Unified Memory and NVLink
- Good for getting up and running quickly
- Can not exploit program knowledge



However, We Do Not Believe These Will be Adequate for Creating Performant Programs

Therefore, We Want to Handle Data Motion Ourselves

- Two different multi-level memory systems coming into production soon.
  - KNL
  - P9 + Volta
- The memory system is expected to continue to evolve.
  - NV-RAM
  - Relative speeds and capacities
- We want to write portable code that stages data into memory efficiently.

Limiting Scope Makes the Problem More Tractable

- Limited to HPC Platforms
  - Clusters
  - Supercomputers
  - Laptops/Desktops (includes Mac and Windows)
  - Includes Non-coherent GPUs/accelerators
- We will hide vendor specific API calls behind a portability layer
  - Need to figure out best practices and where there are common needs
- Focus on C/C++ and Fortran applications
  - This includes the needed performance and debugging tools

What are our concerns and use cases?

Use Cases and Challenges

Multi-Physics and Libraries

Challenge: Codes with multiple physics packages or that call libraries must share resources. They will need to understand the state of the hardware when control is passed, what is allowed and communicate with each other.

Need: Ability to query system for space available and control placement.

TeaLeaf Proxy App

Manual Caching, Blocking and Prefetching

Challenge: Need to prefetch data manually into fast memory before it is needed.

Challenge: Double/Triple buffering?

Need: Ways to understand the relationship between various memory structures bandwidth and latencies, and the processor to optimize motion and place data correctly.

Possibility for specialized hardware to handle buffering schemes?

Partial Migration

Challenge: focus computation on a subset of a large array/data-set

Need: can I move only a *piece(s)* of my larger set into faster memory without needing full buffering/ copies/reallocation?

Codes Run in Varied Ways

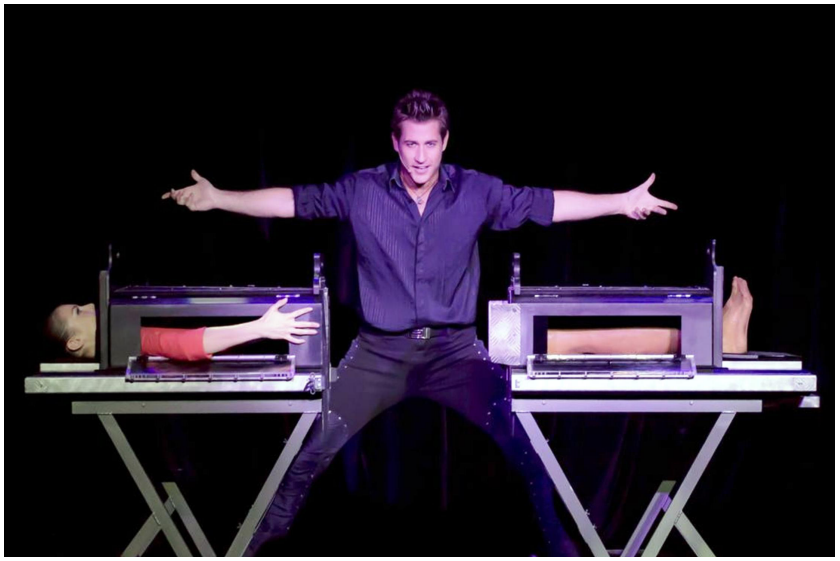
Challenge: Some applications run problems with varied working set sizes and which arrays to put in fast memory will be problem dependent .

Need: A way to specify to the relative priorities that adapt at runtime (when dynamic global view of the requests are known).

Triggered Copy

Challenge: First touch of data triggers a copy of the entire array or class into fast memory.

Need: Support for eviction policies and more complex copy mechanisms if data to be copied does not fit in fast memory.



Asynchronous Task Programming Models/ Runtimes

Long Term Goal: Data movement through the memory system is controlled by a runtime.

Need: Appropriate hooks for runtime. These will likely besimilar to the manual management needs.

Need: Introspection of *why* decisions are made (for analysis/debugging)

There are many common themes in these use cases for what we need.

- Application level programmers think about objects not pages.
  - Someone will need to hide this from them
- Many use cases require the ability to do certain things we can not do on all systems today including:
  - Query for the amount of free space in fast memory
  - Understand the relationship between a cores location on die
  - Control the placement of data within fast memory

Thanks!

- Oliver Perks (AWE)
- Ron Brightwell, Ryan Grant, Si Hammond, Stephen Olivier, Kevin Pedretti, Christian Trott (Sandia)
- Holger Jones, Jeff Keasler, Edgar Leon, Rob Neely, Dave Richards, Tom Scogland (LLNL)
- Ben Bergen, Charles Ferenbaugh, Scott Pakin (LANL)

Lawrence Livermore
National Laboratory

We expect the memory hierarchy to continue to become more complex

- Fast memory will have NUMA access with some cores closer to parts of this memory.
- We want to be able to draw a graph of this memory hierarchy and understand basic measures between memories and cores including:
  - Latency
  - Capacity
  - Bandwidth